COSC460
Low-level Image Segmentation for a Vine Imaging Robot

November 2, 2012

Simon Flowers
sgf24@uclive.ac.nz

Supervisor: Richard Green

Abstract

Image segmentation is an important preprocessing step in most computer vision based applications, as it can significantly reduce future computation in tasks such as object classification. By grouping pixels that are similar with regard to a measure such as colour or position, classification can be performed on a per-segment basis, rather than per-pixel. This research examines several segmentation techniques and evaluates their performance at segmenting the network structure of vine images. Methods described in the literature are selected for comparison based on their performance at segmenting similar structures. The methods examined are k-means clustering, mean-shift clustering, normalised cuts segmentation, quadtree segmentation and watershed segmentation. We evaluate each method against five distinct images, based on their accuracy and efficiency at separating scene components such as vines, posts, wires and background. Evaluation is performed using a boundary-based comparison method to compare segmented images against hand generated ground truths. The clustering methods k-means and mean-shift are found to have the best performance. We propose mean-shift as the most suitable algorithm, due to its ability to produce a dynamic number of segments. We provide reasoning behind the relative successes and shortcomings of each method.

Acknowledgements

Tom Botterill for his extensive help in proof-reading this report and clarifying my understanding of many of the concepts used in this research.

Contents

1 Introduction

Image segmentation is a preprocessing method used in most vision-based applications to separate images into distinct regions. Each region corresponds to a distinct object, or part of an object in the scene. Segmentation is usually applied in order to reduce the complexity of vision tasks later in the processing pipeline, such as object classification [18] and tracking [32]. It reduces the complexity of such tasks by allowing us to use only parts of the image that we are interested in and also allows us to analyse an image at the segment level, rather than on a per-pixel basis [31].

This paper examines image segmentation methods in the context of an autonomous vine imaging robot. Currently, vine pruning is a labour intensive task, so automation of this task using a robot would greatly reduce vineyard labour costs [27]. The robot should move along a row of vines at a constant speed, and be able to determine the best locations to cut the vines, then move a set of shears to those points and execute the required cuts.

In order to determine the best locations to cut the vines, a 3D model of the vine must be constructed from a series of 2D images from multiple cameras, along with a structured light system, as proposed by Botterill et al. [8] [9]. This requires regions of vine to be separated from everything else that appears in the image, which might include posts, wires, string and other objects. With image segmentation as a preprocessing step, this classification can be performed on a per-segment, rather than per-pixel basis, reducing the size of the classification task. Segmentation also helps in reducing the effects of sensor noise. An ideal segmentation would place all distinct components, such as vines, wires, posts and background, into distinct segments. Separation of vine segments is necessary to construct the model of the vine, while separation of wires and posts is also necessary for path planning.

This paper examines several different segmentation algorithms, with the goal of identifying the method best suited to segmenting images of vines. An ideal method would consistently perform well at segmenting images of vines, which primarily consist of many long, thin segments with a complex, networked structure. Research into the segmentation of similar structures provides valuable insight into the characteristics required by a suitable algorithm.

1.1 Research Contributions

A significant portion of this paper is dedicated to the review of related work in the area of segmenting networked structures. Literature that describes image segmentation for a specific application typically provides little insight as to why a particular method was selected in favour of other available methods. We provide a detailed review of methods used in similar applications, and use this as the basis for selecting algorithms to examine.

Research into objective segmentation evaluation and the comparison of different algorithms is usually performed in a general context, as in [35] and [37]. This research presents a novel evaluation of methods in the specific context of segmenting images of vines.

1.2 Report Outline

This paper examines five different segmentation algorithms: k-means clustering, mean-shift clustering, normalised cuts segmentation, quadtree segmentation and watershed segmentation, comparing their effectiveness at segmenting vine images. Chapter 2 provides background on image segmentation in general, followed by background on objective segmentation evaluation techniques. Background on the segmentation of similar networked structures is covered at the end of Chapter 2. Chapter 3 describes each of the five algorithms under evaluation in detail. Chapter 4 describes the method we have used to compare segmentations, followed by the description of our parameter estimation technique. Chapter 4 ends with a presentation of the results, showing the performance of each of the five algorithms when applied to vine images. Chapter 5 contains a discussion of these results, including reasoning behind the successes or shortcomings of each algorithm, as well as proposing ways in which they could be improved. Chapter 6 concludes the paper, relating our findings to the literature in Chapter 2 and discussing plans for future work.

2 Background and Related Work

This chapter describes the motivation for this research, a broad overview of the different types of segmentation algorithms that will be mentioned throughout the report, background on objective segmentation evaluation, and a detailed review of literature that describes the segmentation methods used in similar applications.

2.1 Motivation

The goal of this research is to find a segmentation method suitable for use in an autonomous vine imaging robot. For the robot to be able to make decisions about where to prune a vine, we need to build a 3D model of the vines, in which sections of vine are extracted from the other components in the scene. Once an image of the scene has been successfully segmented, we are able to try to classify segments as belonging to a particular component, such as vine, post, wire, or background. This classification could be performed using methods such as Markov random fields or model-based skeletonisation. The sections that are labelled as belonging to a vine can then be extracted and used to build the 3D model. Without segmentation, this classification would have to be performed on a per-pixel basis, making it a much more computationally intensive task. Perceptual grouping of pixels into segments in this way is also known as "superpixel grouping" [22].

2.2 Segmentation methods

There are many approaches to image segmentation, and there is potential for each approach to produce significantly different results, since each method differs in how it defines a segment. Several common approaches are described below. These broad algorithm types encapsulate most current image segmentation techniques, though there are other methods which have not been examined in this research.

2.2.1 Clustering methods

In general terms, data clustering refers to the act of assigning a set of data into groups, or clusters, based on some measure of similarity [17]. Data clustering can be applied to many problems, including image segmentation. These methods can be applied to image segmentation tasks by clustering pixel data based on information such as spatial position and colour [11]. Pixels of a similar colour that are close together will be labelled as belonging to the same cluster, and the resulting clusters correspond to segments in the image. Specific

examples of clustering methods include k-means clustering and the mean-shift algorithm, both of which are described in more detail in Chapter 3.

2.2.2 Graph partitioning methods

Graph partitioning methods consider the input image as a weighted, undirected graph, where vertices of the graph correspond to pixels and the weights of edges correspond to the level of similarity between the two connected pixels [20]. Segmentation is achieved by finding an optimal partitioning of this graph with respect to some cost function. The location of these partitions represent segment boundaries. In contrast to clustering approaches which perform segmentation by assigning segment labels to individual pixels, these approaches perform segmentation by finding boundaries between segments. The choice of cost function used will determine the final outcome of the segmentation, and the method used to minimise the chosen cost function will impact the computational efficiency of the segmentation algorithm.

2.2.3 Split-and-merge methods

Split-and-merge methods work by repeatedly splitting non-homogeneous regions then merging neighbouring regions with similar characteristics [21]. An example of a homogeneity criterion for the split and merge operations is pixel colour variation. If the variance in pixel colour within an image is above a threshold, then the image is split into equal sized regions. If the colour variance within any of the new regions is above the threshold, then they are further split, and so on. If after the splitting process there are two or more adjacent regions that are below the split threshold, this indicates that most of the variation came from one of the other sub-regions, and the regions with low variation should be merged together. Examples of split-and-merge algorithms are quadtree segmentation and octree segmentation, which involve splitting segments into four and eight subsegments respectively. Quadtree segmentation is used for segmenting 2D images, while octree segmentation is applied to 3D image data, splitting the box formed by the 3D data into eight smaller cubes. Octrees are typically used to segment 3D images from MRI and CT scans commonly used in medical imaging applications.

2.2.4 Region growing methods

Region growing methods use a set of "seed" pixels as the initial regions, then grow each region by adding the neighbouring pixels that are similar in some respect [21]. Region growing methods are analogous to the "merge" operation described in the previous section. The key difference is the need for seed pixels. Initial seed pixel placement can be specified either interactively or automatically, and the locations of these seed pixels can have a large impact on the resulting segmentation.

2.2.5 Model-based segmentation methods

Model-based methods use knowledge of the shape, colour and texture of particular objects to determine the locations of those components in an image.

Model-based techniques are commonly used in medical image processing, where there is a high level knowledge of anatomic structures [29] and common structure between image data for different patients.

Model-based approaches are used for object classification in the next stage of the image processing pipeline for our robot [9].

2.3 Objective Evaluation of Segmentation Results

In order to identify the best method for use in a vine pruning robot, a segmentation evaluation method is needed, so that we can compare the results of different algorithms against each other. In most prior work this evaluation involves comparing a segmented image against some "ground truth" segmentation. Since there is no single interpretation of the ground truth, evaluation of image segmentations in this way is a rather ill-defined problem [37].

Various "unsupervised" evaluation methods, which do not require a ground truth, have been proposed. These methods evaluate a segmentation based on how well it fits a broad set of desirable characteristics such as intra- and inter- region variance and region entropies [38]. Unsupervised methods are primarily used for evaluating general purpose segmentation, where a series of input images are used and the structure of the components in these images is unknown. Unsupervised methods can also be biased, giving unnecessarily high accuracy scores to certain conditions such as over- or under-segmentation [38]. To verify with certainty whether or not a segmentation is accurate, we need to compare it against a hand generated ground truth, which represents the most desirable outcome that a segmentation could produce. For this reason, unsupervised methods are ignored for the remainder of this research.

A ground truth represents the most desirable state of segmentation, and usually needs to be generated by hand. The Berkeley Segmentation dataset [25] contains hundreds of images along with many hand-generated ground truths for each image. The dataset is commonly used in segmentation research. However, since the dataset contains a broad range of images, and we are only interested in segmentation of vine images, images from the dataset are not used here.

After generating our own set of ground truth images, we are able to evaluate segmentation algorithms by comparing segmented images against the ground truth. There are several ways in which this comparison can be performed. Such "supervised" methods fall under three groups. Region differencing methods perform evaluation based on the degree of overlap of segments associated with each pixel in the segmentation and its closest approximation in the ground truth [37]. Boundary matching methods compare segment boundaries between the segmentation and the ground truth, while information based methods compare entropies between the segmentations [37].

Unnikrishnan et al. [37] identify four aspects of a successful segmentation evaluation measure. Firstly, there should not be any degenerate cases in which segmentations that

are not represented by the ground truth produce high accuracy measures. Secondly, no assumptions should be made about region sizes. Thirdly, the measure should account for the possibility of ambiguous segment assignment. For instance the pixels around the edge of an object in the image may be blurry, and could easily belong to either the object itself or the background. Finally, the measure should produce comparable scores, allowing meaningful comparisons to be made between different segmentations of the same image. Unnikrishnan et al. [37] also specify three criteria for a good segmentation algorithm: it should be able to produce images that agree with the ground truth, be able to produce correct segmentations over a range of parameter values, and be able to produce correct segmentations for a range of different images.

2.4 Segmentation of Similar Structures

Vine images consist primarily of long, thin segments with a complex networked structure. For a segmentation algorithm to be suitable for use in our robot, it must be able to reliably segment images with this kind of structure. Identifying or designing a segmentation algorithm for vine segmentation can be aided by looking at prior research into the segmentation of similar structures. We examine several similar structures in the remainder of this chapter, briefly describing the segmentation problem in each application, outlining structural similarities with vines, and describing several methods that have been successful at solving each segmentation problem.

2.4.1 Road network extraction

Road extraction from aerial or satellite images allows for automated creation of up-to-date maps for use in applications such as traffic management, urban planning and navigation [26]. Figure 2.1 shows an example of such an aerial road image.

Figure 2.1: Example aerial image of road network. Source: [15]

Roads are typically long and thin, and form part of a larger network. The structure of the road network in an image can range from a single road, for instance in a remote rural area, to a very complex network of many different roads, as in a densely populated urban area. An algorithm that is successful at segmenting roads in the general case, such that it is able to extract single road segments as well as the complex networks seen in urban environments, would most likely also be successful at segmenting vines, as they share a lot of the same structural properties.

Line extraction and analysis techniques have been successful at extracting simple road segments from rural areas [4]. In this case, roads were mostly homogeneous and did not suffer from shadowing or occlusion. Graph partitioning approaches such as normalised cuts have also been used under similar conditions [3].

Clustering approaches such as k-means [39] and colour channel thresholding [15] have been successful in more cluttered urbanised areas, which contain more complex road networks, though post-processing is required to remove areas that are spectrally similar to the roads, such as parking lots [39].

2.4.2 Fingerprint segmentation

Fingerprint segmentation removes noise in fingerprint images, allowing fingerprint features to be more easily extracted [19]. The useful information in a fingerprint image lies in the foreground, so a fingerprint segmentation algorithm aims to separate foreground and background. An example of this can be seen in Figure 2.2. Segmentation forms the first step in a fingerprint identification system.

Figure 2.2: Example of fingerprint segmentation. Top row: original images. Bottom row: segmented images. Source: [19]

While fingerprints have a similar branched structure to vines, the fingerprint segmentation techniques that we are aware of are only interested in segmenting the region of the image containing the fingerprint, not the raw network structure of the fingerprint itself. Fingerprint images generally lie in a greyscale or monochromatic colour space, whereas a vine segmentation algorithm should be able to make use of full RGB colour information. Finally, the level of noise in the background of a fingerprint image is likely to be much lower than that of an image of a vine. For example, in a fingerprint image the background might contain a few areas of smudging. In a vine image, the background might contain thousands of pixels making up posts, wires or other random noise. Nevertheless, fingerprints are structurally similar to

vines, so fingerprint segmentation methods may still be of use in the vine pruning robot.

Fingerprint segmentation methods make use of pixel colour information, for example segmenting by texture using local grayscale variance [19]. This ties in well with split-and-merge techniques where the split and merge thresholds are based on colour variance. Neural networks and gradient based approaches such as Gabor filters have also been used [19].

2.4.3 Brain segmentation

Brain segmentation is an important processing step in clinical diagnostic tools such as tumor and necrotic tissue detection [5]. Segmentation accuracy is extremely important for correct diagnoses, so techniques must be highly robust. Segmentation is also useful in separating different brain structures such as grey matter, white matter and cerebrospinal fluid, which can be used to measure brain development or compare brain images from different patients. The boundaries of each of these structures are similar to vines in that they are complex and somewhat branched, as can be seen in Figure 2.3. These boundaries can be weak, and they usually appear in fairly predictable areas across healthy patients. The level of noise in MRI data can be high, causing reliable, unsupervised segmentation to be a difficult task [5]. Medical imaging systems such as MRI and CT scans also have the advantage of having the full 3D information of the structures in the brain, avoiding the need to account for overlapping components.

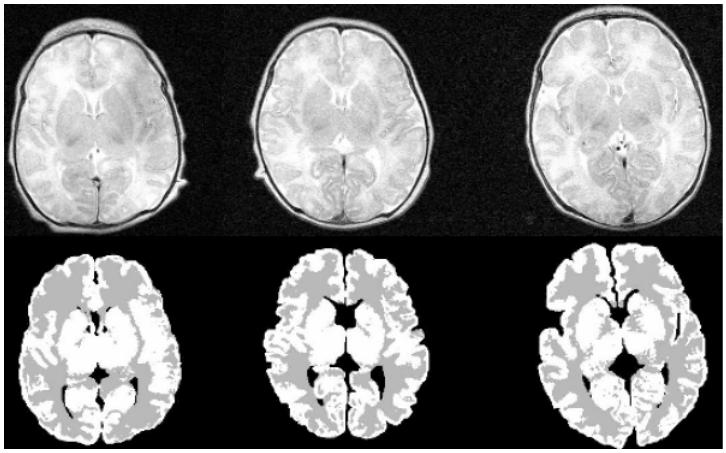


Figure 2.3: Example of source images (top row) and segmented brain MRI images (bottom row) Source: [34]

Because tissue regions occur at fairly similar locations across patients, it is easy to come up with statistical models for use in segmentation, making model-based approaches such as Markov random fields useful here [5]. Clustering approaches such as k-means [33], and region growing methods [5] have also been used.

2.4.4 Artery/vein separation

Visual separation of veins and arteries from other structures in medical images is useful in diagnosing various diseases, studying malformations in arterial or venous structures, and for surgical planning [36]. Separation of these structures from other data in the image allows 3D models of the structures to be built, which can then be more easily examined by a doctor. An example reconstruction of the blood vessels in the lungs is shown in Figure 2.4. It may also be necessary to separate the veins from the arteries, for instance by rendering each type in a different colour in the reconstruction.

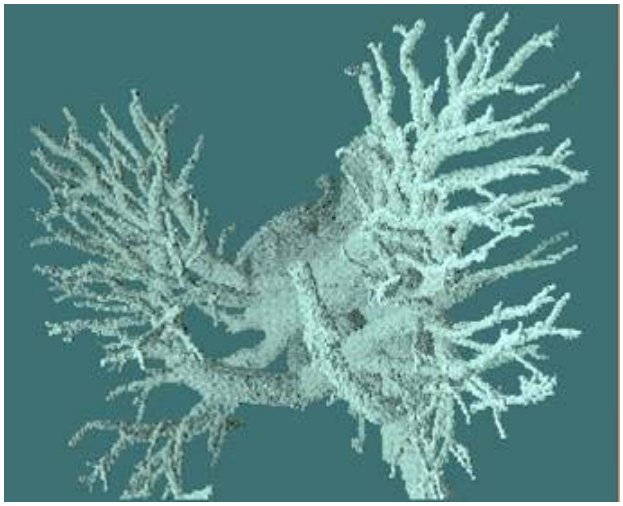


Figure 2.4: Example of segmented bronchial blood-vessel tree of the lungs by region growing and merging. Source: [13]

The complex network structure of veins and arteries in the human body is analogous to the structure of vines, however the medical images have the advantage of full 3D information, avoiding the complications that arise from overlapping components.

Region growing methods and local colour thresholding techniques have been used successfully in this area [13] [36].

2.4.5 Vine pruning

Research into vine image processing has been performed in the past. McFarlane et al. [27] describe the image analysis techniques used in making measurements such as direction, diameter and length of vine segments, which are used for prune location decision making. They achieve segmentation through a simple approach of colour thresholding and size filtering, but they do not provide any information on why this particular technique was chosen. As such, it was likely due to its simplicity that this technique was selected, and it is not stated whether alternative methods were considered. The authors mention limitations such as loss of grey level information and difficulty in resolving overlapping and adjacent vines. As this paper was published in 1997, it makes sense to consider different approaches given the greater processing power and wider knowledge base we have today. Similarly, Naugle et al. [28] describe the

image processing required in making a robot follow vine cordons, which requires separation of vines from background. They use grey level thresholding and resolution reduction to achieve this. Spatial reduction would result in a loss of structural information, so cannot be considered for use in our robot. Their paper was published in 1987, and resolution reduction was used to keep the size of the image to be processed low. Again it makes sense to reconsider their segmentation technique given recent advances in the field and greater computational power.

2.4.6 Segmentation of parchment scrolls

Many historical parchment scrolls cannot be physically unrolled as this could damage or destroy the information written on the scroll. With the use of X-ray microtomography (XMT), a digital 3D image of the scroll is generated. This model can then be "virtually unrolled": ink on the surface of the parchment can be detected and the text retrieved, without the risk of physically damaging the scrolls [30]. Figure 2.5 shows an example of a single slice of the XMT scan with a computer generated cutaway view of the scroll.

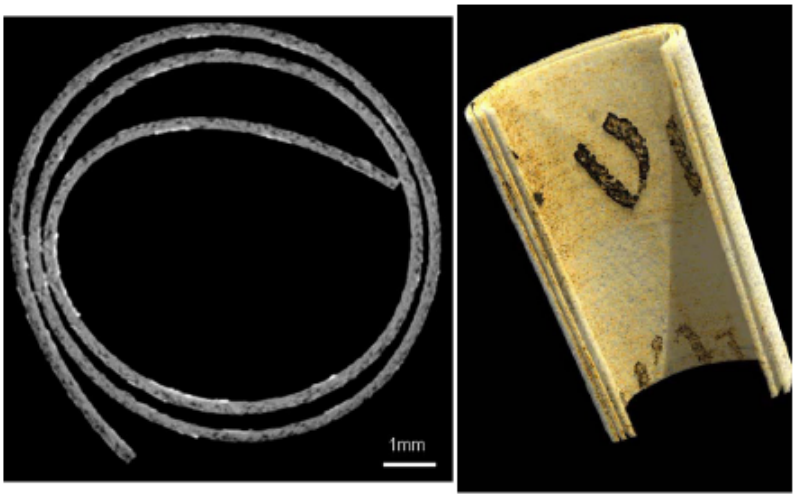


Figure 2.5: Left: cross-sectional XMT slice of a parchment scroll, contrast enhanced to show ink on the surface. Right: virtual cutaway view. Source: [30]

Slices of parchment scroll in the 3D model are long and thin, and in some places joined together, forming networked structures similar to vines. However, the scrolls do not possess structures of the same level of complexity. Samko et al. [30] use a graph partitioning approach for separating segments of parchment from background noise in the XMT. Their approach is a modified version of Boykov and Jolly's "Graph Cuts" algorithm [10] that uses additional shape information such as parchment thickness. Postprocessing techniques are required to separate layers of parchment that are stuck together in the segmentation.

2.4.7 Summary of method types used to segment similar structures

Clustering methods have found use in complex urban road segmentation [39] and brain segmentation [33]. Region growing methods have been used in vein/artery segmentation [13] and brain segmentation [5]. Graph-based approaches have been used for simple road network extraction [3] and parchment scroll segmentation [30], while colour variance thresholding and local colour thresholding techniques (such as those utilised by split-and-merge methods) were used in fingerprint segmentation [19], vein/artery segmentation [36] and vine segmentation [28] [27].

3 Segmentation Algorithms

Based on the segmentation methods used in prior work, we selected five methods for evaluation: k-means clustering, mean-shift clustering, normalised cuts segmentation, quadtree segmentation and watershed segmentation. The clustering methods were chosen due to the use of clustering methods in complex urban road network extraction [39] and brain segmentation [33]. Normalised cuts is a graph-partitioning method and was chosen due to its use in simple road segmentation [3] and the success of a graph-based approach for parchment scroll segmentation [30]. Quadtrees was selected due to the use of variance thresholding techniques in fingerprint segmentation [19], while watershed was selected due to the success of region growing methods in medical imaging applications such as brain segmentation [5] and vein/artery segmentation [13] [36]. Each of these algorithms are described in more detail in the remainder of this chapter.

3.1 Mean-shift Segmentation

The mean-shift algorithm is a non-parametric data clustering procedure for finding the modes of a probability density function (PDF) [14]. Mean-shift considers image pixels as 5-vectors, with elements in the vector corresponding to red, green, and blue colour values, as well as spatial x and y position values. The algorithm is applied to image segmentation by finding modes of the underlying probability density function of an image, then grouping modes that are close to one another in regard to their colour and position [16]. The implementation used here requires two inputs: the "spatial radius" and "colour radius". These specify the maximum difference in position and colour that two modes must be within in order to be placed in the same segment.

For each pixel in the image, we define a window around that pixel and calculate the "mean-shift vector". We then shift the window by this vector and repeat until convergence. The point of convergence is the nearest mode of the underlying PDF for that pixel. For each mode, the pixels that converge to that mode are assigned to the same cluster. Clusters that are within the spatial and colour radius's distance from one another are grouped together, and these groups correspond to segments in the final image [12].

3.2 k-means clustering

k-means is another clustering method that assigns image pixels to one of k clusters based on their spatial and chromatic distance from one another. Like in the mean-shift clustering algorithm, each pixel is represented as a 5-vector, corresponding to that pixels red, green,

blue, x, and y components. Our implementation allows the importance of spatial information relative to colour information to be varied. The algorithm used to classify pixels to clusters is known as Lloyds algorithm [24] and works as follows:

1. A centroid for each cluster is chosen randomly or by some heuristic, resulting in k centroids. (We used random centroid placement, as this noticeably reduced algorithm run-time while having little to no impact on accuracy)
2. Each pixel is associated with the centroid to which it is most similar (i.e. has the smallest spatial/chromatic distance)
3. Centroids are recalculated for each cluster found in the previous step
4. Steps 2 and 3 are repeated for a fixed number of iterations or until the movement of centroids is below some value (these are specified as inputs to the algorithm)
5. Once the algorithm has assigned all pixels to a cluster, each pixel is coloured based on the cluster it is assigned to.

3.3 Watershed Segmentation

Watershed based segmentation considers an image as a topographical map, based on image gradient. "Water" placed in a regional minimum floods regions, and barriers are built when water from neighbouring sources meet. The resulting barriers correspond to segment boundaries. Figure 3.1 illustrates this concept. Each catchment basin (CB) corresponds to a segment in the resulting image.

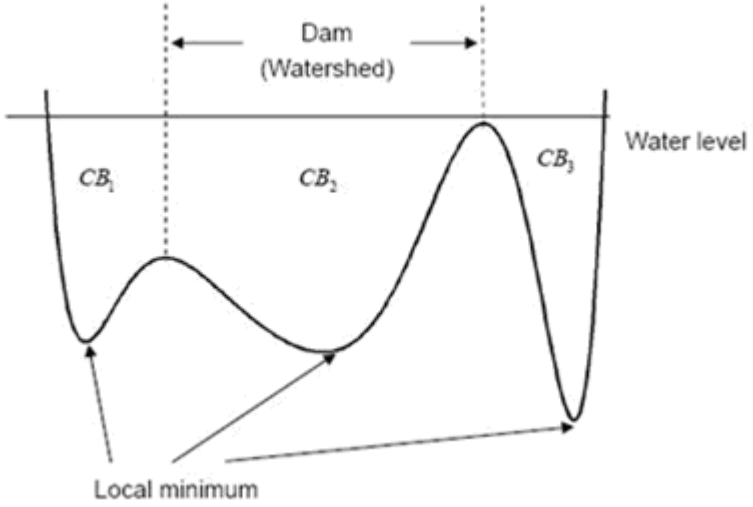


Figure 3.1: Visualisation of watershed segmentation. Source: [7]

The implementation that has been selected is Meyer's Flooding Algorithm [6], which can be described as follows:

1. A set of seed pixels (markers) are chosen, which represent the pixels where flooding will begin from, and each marker is assigned a label. The locations of these markers can be chosen by hand or determined automatically. In our case markers are simply placed in a regular grid, where the size of the grid is specified as an input parameter.
2. Neighbouring pixels of each marker are assigned a priority based on their intensity and added to a priority queue.
3. The pixel with the highest priority is extracted from the queue. If all of its neighbours have the same label, then this pixel is assigned the same label. All unmarked neighbours that are not in the queue are assigned a priority and added to the queue.
4. Repeat the previous step until the queue is empty
5. The remaining non-labelled pixels are the watershed lines, and correspond to segment boundaries.

3.4 Normalised Cuts Segmentation

Normalised cuts [20] is a graph-based approach, considering an image as a weighted, undirected graph, $G = (V, E)$, where vertices V correspond to pixels, and the weightings of edges E correspond to a measure of similarity between pixels. Image segmentation is analogous to finding the optimal partitioning of this graph, where the locations of the partitions represent segment boundaries. A partition of G into two subgraphs A and B has a cut cost function:

$$cut(A,\ B) = \sum_{u \in A, v \in B} w(u, v) \tag{3.1}$$

where $w(u, v)$ is the weight of the edge connecting vertices u and v.

This cut function favours partitioning small, isolated sets of nodes, so a "normalised cut" function is used:

$$Ncut(A,\ B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{3.2}$$

$$assoc(A,\ V) = \sum_{u \in A, t \in V} w(u, t) \tag{3.3}$$

where $assoc(A, V)$ is the total connection from nodes in the subgraph A to all nodes in the graph. We estimate subgraphs A and B such that

$$(A, B) = argmin \quad Ncut(A, B) \tag{3.4}$$

Small, isolated sets of nodes will not have a small $Ncut$ value, because the cut value will be a large percentage of the total connection from that set to the others.

To find the minimum $Ncut$ value, we consider the problem in the form of solving a generalised eigensystem. We construct a symmetric matrix W with values $W_{ij} = w(i, j)$ and a diagonal matrix D with elements d on the diagonal where $d(i) = \sum_j w(i, j)$.

Solving $(D - W)y = \lambda Dy$, where y is a real number, for the eigenvector with the second smallest eigenvalue, gives us the optimal bipartitioning of G [20]. We can now describe the normalised cuts segmentation algorithm as follows:

1. Given an image, set up a weighted graph $G = (V, E)$ and set the weight of the edge connecting two nodes to a measure of similarity between the two nodes.
2. Solve $(D - W)y = \lambda Dy$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph
4. Subdivide and recursively repartition if the desired number of segments has not been reached.

3.5 Quadtree Segmentation

Quadtree segmentation is traditionally a split-and-merge method which segments an image by recursively dividing it into four equal sized blocks. If the variation in pixel colour within a segment is below a specified value, then that segment is marked as a leaf in the quadtree, and is not divided any further. Once all segments have been marked as leaves, the segmentation is complete. Normally this would be followed by a "merge" operation in which neighbouring segments which have little variation between them would be merged back together, however the implementation used here only consists of "split" operations.

3.6 Implementation

The k-means clustering, mean-shift clustering and watershed segmentation methods were implemented using the OpenCV 2.3 C++ libraries. For normalised cuts segmentation we used Shi's MATLAB implementation [1], and for quadtrees we used an open source MATLAB implementation [2].

4 Comparing Segmentation Methods

To evaluate the performance of the algorithms described in Chapter 3, the segmented images are compared against a ground truth segmentation. Ground truth images represent the most desirable segmentation state of a particular image. In our case, the ground truth is an image in which each distinct scene component, such as vines, posts, wires and background, are placed in separate segments.

We constructed ground truths for each of the 160x160 pixel images in Figure 4.1 and 4.2. The ground truths are shown in Figure 4.3. We selected images that covered a wide range of structural complexity, ranging from a single vine to multiple overlapping vines, wires and posts. Two different types of vine images were used as inputs. The images with green backgrounds were from an earlier camera and lighting setup, while the images with blue backgrounds were taken with a more advanced setup. Using images from two different setups helped to support the generalisability of evaluation, as the performance of a suitable algorithm should be minimally impacted by such variability. Additionally, by using a set of images over a wide range of structural complexity, we further support generalisability, as a successful algorithm should perform well against a wide range of possible image structures.

In generating the ground truths, sometimes it was unclear as to exactly which segment pixels on object boundaries should be placed. There is a band of several pixels on the border of each object where the colour of the foreground object blends into the colour of the object behind it, and it is unclear as to on which pixel the actual change from foreground to background occurs. This is illustrated by Figure 4.4. As such, it is left up to the person creating the ground truth to decide which of these border pixels belong to which segment,

(a) Single vine with tendrils | (b) Overlapping vines and wires | (c) Overlapping vines and wire | (d) Overlapping vines and wire | (e) Many overlapping vines, tendrils and a section of post

Figure 4.1: Training images (a)-(e): the 160x160 pixel training images

(a) Multiple overlapping vines with wire

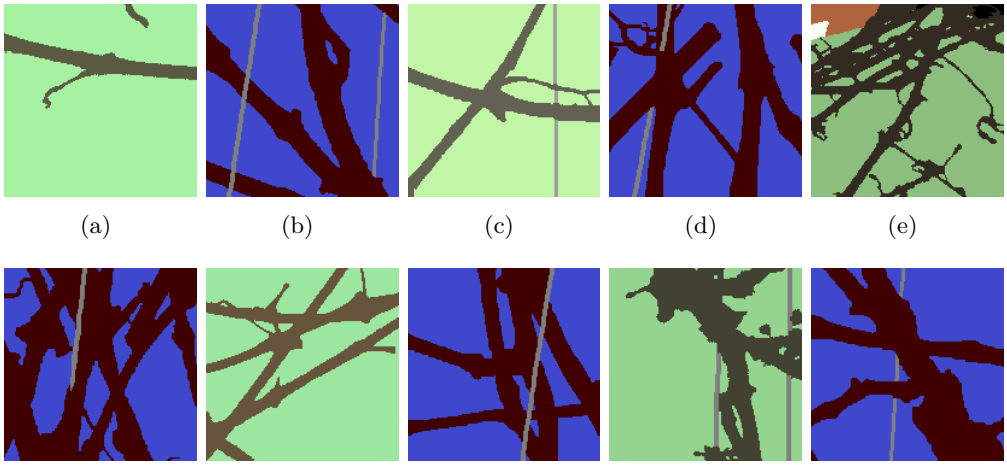
(b) Multiple overlapping vines

(c) Multiple overlapping vines with wire

(d) Trunk, vines and wires

(e) Trunk, vines and wire

Figure 4.2: Input images (a)-(e): the 160x160 pixel input images



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.3: Ground truths for training and test images 4.1 and 4.2

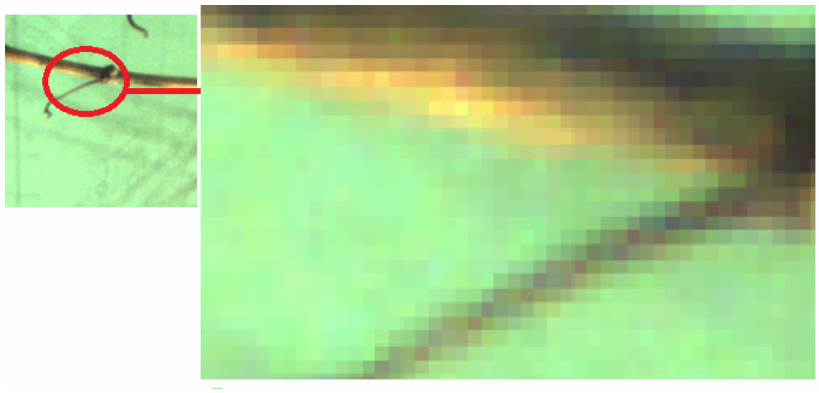


Figure 4.4: Illustration of the blurring of colours around object borders

meaning there are many possible valid ground truths for a given image. The images with blue backgrounds have a sharper contrast between foreground objects and the background due to the more advanced camera and lighting setup, making ground truth generation relatively easier.

For our purposes, over-segmentation is preferable to under-segmentation. Under-segmentation refers to the case when pixels belonging to different objects are placed in the same segment, while over-segmentation is when a single object is split into several segments [23]. Under-segmentation will result in fine structural detail being lost as a result of the segmentation, reducing the accuracy of any 3D model built afterwards. Over-segmentation may result in more segments than are necessary, but retains fine structural detail.

We compared segmentations to their respective ground truths using the boundary-based comparison method outlined in Algorithm 1. Put simply, this algorithm checks for the location of segment boundaries in the ground truth, then checks for a boundary in the corresponding location in the segmented image. Accuracy is then reported as the number of boundary pixels in both the ground truth and the segmented image over the total number of boundary pixels in the ground truth. For the purpose of this comparison, we consider segment boundaries in both the segmented image and the ground truth to be anywhere that two neighbouring pixels are different colours. To account for the ambiguity of segment assignment for pixels on the borders of objects, we allow a small margin of error, allowing the precise location of the border in the segmentation to be anywhere within a three-pixel band. The size of this margin was set to three pixels, as it was observed that there are usually approximately three pixels in any given direction around the border of an object that have some ambiguity as to which segment they should be assigned.

Our segmentation evaluation method satisfies three of the four conditions for a successful segmentation evaluation measure [37]: it makes no assumptions about segment size, it accounts for possible ambiguities in segment assignment by checking for boundaries within a band of three pixels, and it produces comparable scores, allowing meaningful comparisons to be made between segmentation results. Our measure does however fail to account for degenerate cases, as there are segmentations that will produce high accuracy scores without being well represented by the ground truth. An example of this would be a segmentation in which every pixel in the image is a different colour. Since we consider a segment boundary as

anywhere that two neighbouring pixels have different colours, this would cause every pixel to be surrounded by a distinct border, producing 100% accuracy under our measure. This can be accounted for by manually examining individual segmentations and making sure there is a level of correspondence between segmentation and ground truth.

Algorithm 1 Evaluation Algorithm

```
for all pixels in the ground truth image do
    if current pixel is a different colour to the pixel to the right then
        mark this pixel as having an edge to the right
    end if

    if current pixel is a different colour to the pixel below then
        mark this pixel as having an edge below it
    end if
end for

for all pixels in the segmented image do
    if the corresponding pixel in the ground-truth image has an edge to the right or below
(or both) then
        check whether there is an edge in that direction in the segmented image
    end if
end for

Report segmentation accuracy as (number of edge pixels in both the segmentation and
ground truth / total number of edge pixels in the ground truth)
```

Note that Algorithm 1 only checks for edges below and to the right of the current pixel in the ground truth, as checking for edges in all directions would lead to redundant comparisons. That is, an edge to the right of one pixel can be equivalently represented as an edge to the left of another pixel.

The comparison algorithm was implemented using the OpenCV 2.3 C++ libraries and was executed using a laptop with a 2.13GHz Intel Core i3 processor and 4GB of RAM.

The images in Figure 4.1 were used for parameter training, which will be described in more detail in the following section. The images in Figure 4.2 were used as test images for the overall evaluation.

4.1 Parameter Estimation

Each of the five methods described in Chapter 3 requires a set of parameters in order to run. These inputs can have a large impact on the resulting segmentation, so it is important that appropriate values are chosen. A successful algorithm should be able to produce good results over a range of parameter values, and should also be able to produce consistent results using the same parameter values across multiple images [37].

In order to estimate the best parameter values, we use a brute-force method which iter-

ates over a range of values for each parameter, producing a distinct segmentation for each parameter set. Each of these segmentations were then evaluated by comparing them against their ground truth. This process was applied to the five training images in Figure 4.1. The set of parameter values that produce segmentations that satisfy a cost function were selected for general use.

4.1.1 Cost Function

In determining the best parameter value set to use, there is often a trade-off between the number of segments produced and the accuracy of the segmentation. The goal of image segmentation in our case is to reduce the amount of computation required for future vision tasks such as object classification. Usually an accurate segmentation requires many segments, meaning the amount of future computation avoided as a result of the segmentation will be less. Conversely, a segmentation with less segments will most likely be less accurate, as detailed border information tends to be lost. Having fewer segments will reduce the amount of future computation by a larger amount. At one extreme, we will always be able to achieve a 100% accurate segmentation by labelling each individual pixel as a unique segment. This is equivalent to not performing a segmentation at all. On the other hand, we can minimise the number of segments by assigning all pixels in the image to a single segment, but this will result in a 0% accuracy measure.

To allow for reliable construction of a 3D model of the vines, the accuracy value must be very high in order to minimise the amount of important information lost as a result of the segmentation. On the other hand, the number of segments must be significantly less than the number of pixels in the original image in order to justify the use of segmentation as a preprocessing step by reducing the amount of future computation. With this in mind, we impose constraints on the number of segments and the accuracy of segmentation to allow us to select parameter values for general use: a segmentation should aim for at least 98% accuracy. The parameter values that achieve this accuracy in all training images, such that the average segment count is minimised and is below 12,800 will be selected for general use. If a 98% accuracy cannot be achieved, we use the values that produce the highest accuracy value while maintaining a segment count of less than 12,800 (half the maximum of 25,600 for our 160x160 pixel images).

To aid in deciding the best parameter values, accuracy is graphed against the number of segments produced for each set of parameter values, using the training images in Figure 4.1. These are shown in Figures 4.5 - 4.9. Note that the result sets for each image within each of the graphs are of a similar shape to one another, with all graphs except normalised cuts converging on an accuracy value close to 100% as the number of segments increases. Mean-shift and k-means converge on 100%, while quadtrees and watershed converge to just below 95%, though quadtrees reaches this convergence at a number of segments higher than our limit of 12,800. The convergence indicates a level of generalisability for estimated parameters.

A preliminary parameter estimation was performed using only two training images: Figure 4.1 (a) and (c). Increasing the training sample set to include all images from Figure 4.1 produced very little change to the estimated parameters. This further supports the generalisability of our parameter estimation, and shows that results have fairly low sensitivity to

changes in parameter values.

The parameter names and estimated best values for each method are shown in Table 4.1.

Segmentation Method	Parameter Name	Value
k-means	k spatial weighting	5000 0.1
Mean-shift	spatial radius colour radius	15 10
Watershed	grid spacing	3x3
Normalised Cuts	# segments	100
Quadtree	colour std. dev. within segment	0.03

Table 4.1: Required parameters for each method, and the estimated best value

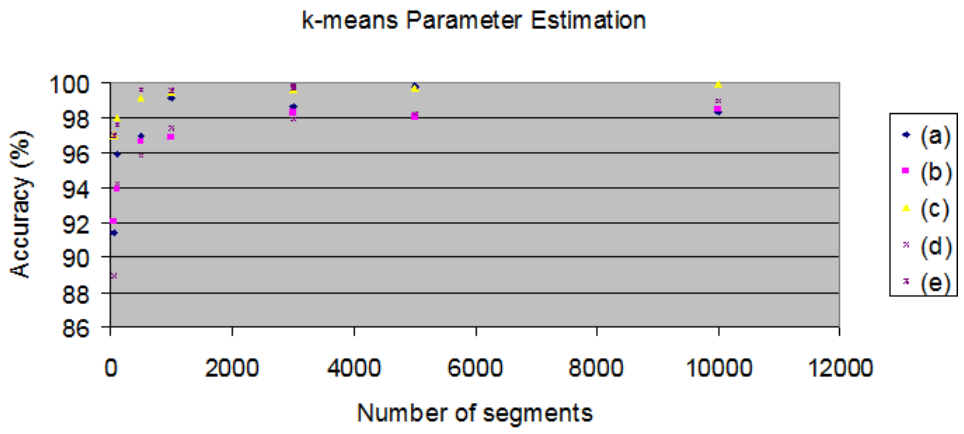


Figure 4.5: Plot of accuracy vs. number of segments on images Figure 4.1 (a)-(e) using k-means clustering

4.2 Results

Using the parameter values in Table 4.1, we evaluated each method for its suitability for use in a vine pruning robot. Each method is run on the five images (a)-(e) in Figure 4.2 and evaluated against the corresponding ground truths. The results are shown in Table 4.2

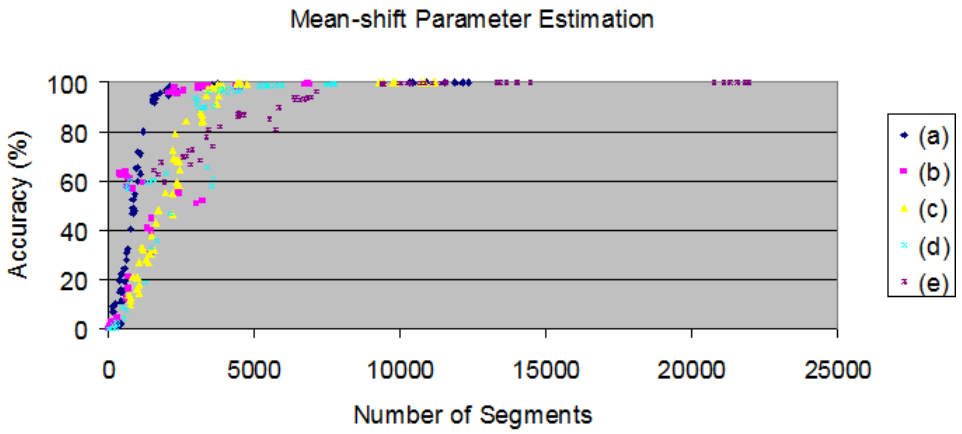


Figure 4.6: Plot of accuracy vs. number of segments on images Figure 4.1 (a)-(e) using mean-shift clustering

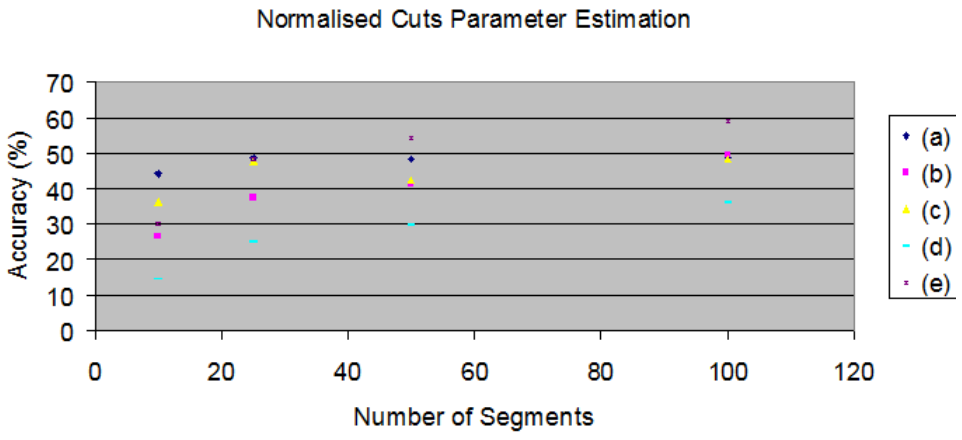


Figure 4.7: Plot of accuracy vs. number of segments on images Figure 4.1 (a)-(e) using normalised cuts segmentation

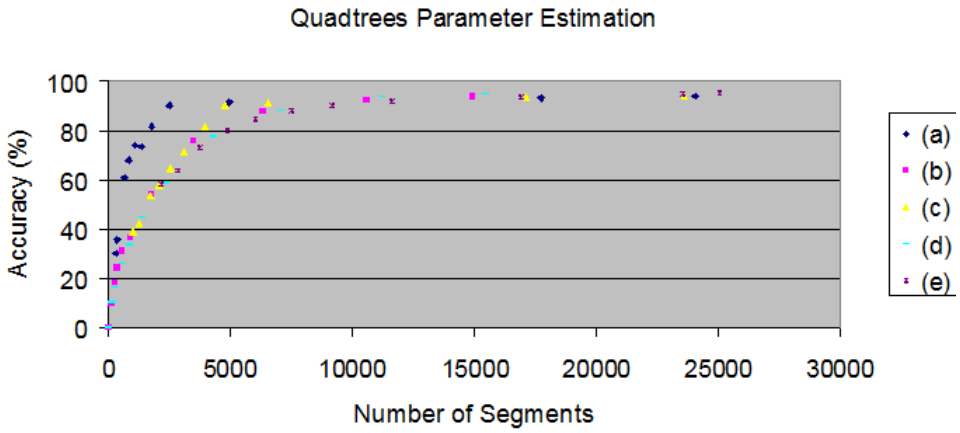


Figure 4.8: Plot of accuracy vs. number of segments on images Figure 4.1 (a)-(e) using quadtree segmentation

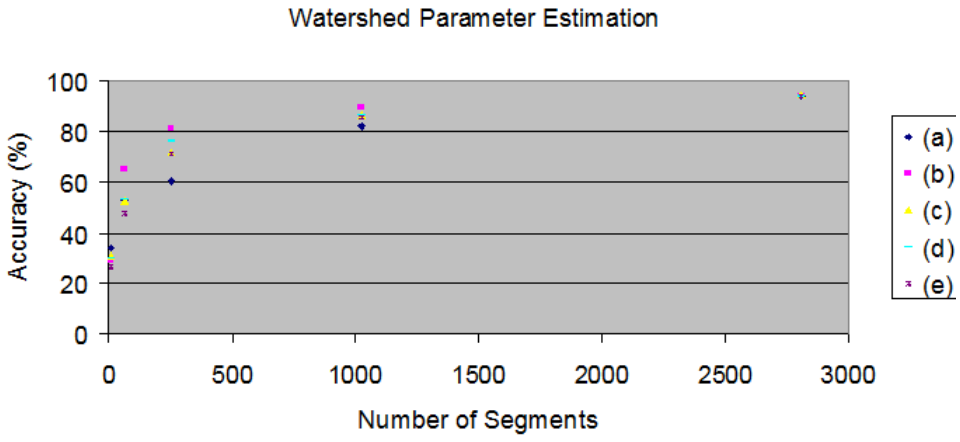


Figure 4.9: Plot of accuracy vs. number of segments on images Figure 4.1 (a)-(e) using watershed segmentation

Segmentation Method	Image (Figure 4.2)	Accuracy (%)	Number of segments
k-means	(a)	98.65	5,000
	(b)	100.00	5,000
	(c)	99.37	5,000
	(d)	99.53	5,000
	(e)	98.45	5,000
Mean-shift	(a)	97.70	7,744
	(b)	99.88	8,526
	(c)	98.87	3,846
	(d)	99.53	8,488
	(e)	99.30	4,330
Watershed	(a)	92.89	2,809
	(b)	93.79	2,809
	(c)	95.40	2,809
	(d)	93.75	2,809
	(e)	92.74	2,809
Normalised cuts	(a)	36.22	100
	(b)	74.57	100
	(c)	63.01	100
	(d)	64.45	100
	(e)	41.81	100
Quadtrees	(a)	80.61	6,277
	(b)	91.28	5,835
	(c)	76.24	3,091
	(d)	89.97	6,729
	(e)	78.59	3,024

Table 4.2: Segmentation accuracy and number of segments produced by each method on each of the input images in Figure 4.2

.

5 Discussion

In this chapter we discuss the results obtained in Chapter 4, as well as providing reasoning behind the relative successes and shortcomings of the various algorithms examined.

5.1 Mean-shift Segmentation

The mean-shift segmentation algorithm was very successful. The algorithm achieved a mean accuracy score of 99.06% ($\sigma = 0.84\%$), in a mean of 6,587 segments ($\sigma = 2309.64$ segments) over all test images. The parameter estimation stage showed that there were many sets of parameter values that produced a close to 100% accurate segmentation, though some cases required significantly more segments to achieve this. Even for parameter values that achieved relatively low accuracy scores under my comparison method subjectively looked to produce very good segmentations, and managed to achieve this in very few segments.

The success of mean-shift here is due to the fact that the algorithm is minimally impacted by the shapes of structures in the image.

The number of segments produced by mean-shift is dynamic, which is advantageous as we do not have to determine the number of segments to use in advance. This point is expanded in the discussion of k-means.

5.2 *k*-means Clustering

The k-means algorithm performed similarly to mean-shift. Both are clustering algorithms, so the fact that they produce similar results is unsurprising. k-means achieved a mean accuracy measure of 99.20% ($\sigma = 0.64\%$), in 5,000 segments.

Unlike mean-shift, k-means requires a static k value as an input parameter. This is a disadvantage as it has the potential to limit segmentation accuracy by restricting itself to too few segments. On the other hand it could result in an unnecessarily high number of segments. For instance, the value of k used here was 5,000 for a 160x160 image, as that was the lowest value that achieved a 98% accurate segmentation over all training images. It is possible that higher accuracy could have been achieved in the test images using a higher k value, or that some images could have been segmented with a lower k value while still achieving satisfactory accuracy levels.

Though algorithm run-time was not incorporated into our evaluation, the implementation used here took a long time to execute, on the order of several minutes for a k value of 5,000. The algorithm would have to be heavily optimised in order to be used in our robot, which

must process many frames per second.

5.3 Watershed Segmentation

Watershed segmentation performed satisfactorily, though not as well as the two clustering methods. Watershed had a mean accuracy score of 93.71% ($\sigma = 1.06\%$), in 2,809 segments. The naive approach to seed positioning was a heavily limiting factor. Seed points are simply placed in a regular grid, with the spacing of this grid as an input parameter. The minimum possible grid spacing is 3x3. If this is reduced to 2x2, then we have segment boundaries surrounding segments of a single pixel, making the segmentation redundant.

An ideal watershed segmentation would take n seed points, where n is the number of distinct components in the image. Each seed point would be positioned on a different component, and the watershed lines would form around the edges of those components. To achieve this ideal seed positioning we would need some way of knowing where the distinct components are in advance. Knowing these component locations would require prior knowledge of the structure of the image, which we do not have without prior segmentation and classification.

5.4 Normalised Cuts Segmentation

The normalised cuts segmentation algorithm performed very poorly and had the worst accuracy of the algorithms examined, with a mean accuracy score of 56.01% ($\sigma = 16.26\%$), though it used the lowest number of segments (100).

Regions of background are always split into segments of similar size, despite most background pixels being very similar in colour. The success of the clustering methods indicates that a successful method should be able to produce segments ranging from only one or two pixels, for instance around object borders, up to segments containing hundreds of pixels, such as regions of background. This method is particularly poor at correctly segmenting very fine structure such as vine tendrils and thin vine segments. This is due to the "normalised" aspect of normalised cuts, where small isolated segments have a lower priority in the $Ncut$ cost function, and are usually ignored. This could possibly have been improved by reducing the weighting of the normalisation component of the $Ncut$ cost function. This algorithm requires the number of segments n as an input, but can only handle relatively small values due to its computational complexity. Because of this, we use a segment count value of 100, which is very low compared to the several thousand segments produced by the other methods. However, the parameter estimation step showed that increasing the segment count had little effect on the accuracy score. Doubling the number of segments from 50 to 100 only yielded around 5% higher accuracy, indicating that there may be little to gain by improving the algorithm to use a higher segment count. The high standard deviation for accuracy (16.26%) shows that accuracy is very inconsistent across different images.

In the implementation used here, eigensystems, used in the normalised cut cost function, are calculated for each segment. This is highly computationally expensive, causing the algorithm to be very slow even for just 100 segments, taking several minutes to complete. Additionally, the maximum number of segments that we can use is low due to the high memory requirements of the implementation used here.

5.5 Quadtree Segmentation

Quadtree segmentation performed the second worst of the five methods under evaluation, with a mean accuracy value of 83.34% ($\sigma = 6.85\%$) and a mean segment count of 4,991 ($\sigma =$ 1793.45 segments).

This algorithm examined used only "split" operations, with no "merge" component. This raised the number of segments in the final image, meaning we reached our limit of 12,800 segments with a low accuracy value. The parameter estimation step showed that using a low variance threshold produced segmentations with an accuracy close to our aim of 98%, although this required a very high number of segments, far above our limit of 12,800. This could possibly be countered by the introduction of a merge step. Quadtrees has the disadvantage of having to split right down to a single pixel to correctly segment diagonal lines, further raising the number of segments. Due to the low accuracy of quadtree segmentation, low level detail of the structures in the image, such as small vine edge discontinuities, are lost, as are areas of fine structure such as tendrils, thin vine segments and sections of wire.

6 Conclusion and Future Work

We found mean-shift clustering to be the algorithm best suited to segmenting the complex, thin, networked structure of vine images. The two clustering methods, mean-shift and k-means, were found to be the most successful algorithms for segmenting the distinct components of vine images, with both algorithms achieving around 99% accuracy on all five test images. Mean-shift has the advantage that the number of segments in the resultant image is dynamic, while k-means requires a fixed number of clusters (k) to be specified as an input parameter. Using a fixed k value could potentially limit segmentation accuracy or result in an unnecessarily high number of segments. For this reason, we propose mean-shift as the most suitable algorithm for segmenting the distinct components of vine images.

Clustering methods previously found use in applications such as complex road network segmentation [39] and brain segmentation [5]. Complex road networks are one of the structures most similar to vines of those reviewed in the literature, and the success of clustering methods demonstrated here reflects this. Medical applications such as brain segmentation require a very high degree of accuracy, and the high accuracy of clustering methods here is supported by their use in brain segmentation applications.

Variance thresholding techniques, like those used in the quadtree segmentation algorithm, were used in fingerprint segmentation [19]. Fingerprint segmentation only aims to segment foreground from background, rather than the network structure of the fingerprints themselves, so fingerprint segmentation is a less relevant structure than the others that were reviewed. The quadtree segmentation implementation used here performed relatively poorly, reflecting this lower relevance.

Region growing methods were used in both of the medical imaging structures we reviewed (brain segmentation and vein/artery segmentation). As previously discussed, these applications require a high level of accuracy, indicating that region growing methods, such as watershed segmentation, should have had accurate results. While the accuracy measures for watershed were reasonably high, the algorithm was outperformed by the clustering methods, and the accuracy score was never able to reach our desired mark of 98%. Our reasoning for the successful use of region growing methods in medical applications, and unsuccessful performance of watershed segmentation for vine images, is due to seed pixel placement. In medical images, the approximate locations of certain structures such as grey and white matter in the brain, or major arteries, lie in relatively similar positions across patients, meaning there is prior knowledge of where seed pixels should be placed. Vine images have no such predictable structure, so we have no prior knowledge of where to place seed points. Future work could utilise watershed segmentation as part of a hybrid method, where a fast segmentation is performed with little emphasis on accuracy, then using the resulting segment centroids as seed

points for the watershed algorithm.

Graph based approaches were used in segmenting simple road network structures and segmenting parchment scrolls. These structures do not possess the complex network structure of vines, and are instead made up of only simple networks of long, thin segments. From these previous applications of graph based approaches, and our observations of normalised cuts segmentation, graph based approaches are well suited to segmenting structures with large segment boundaries, but are not as capable of segmenting smaller, irregularly sized structures. This contributed to the significantly inferior performance of normalised cuts here, and makes normalised cuts an unsuitable algorithm for our end-system use.

The three less successful algorithms (normalised cuts, quadtrees and watershed) could be adapted to better suit our purposes. Specifically, the computational complexity of normalised cuts could be reduced, allowing a higher number of segments to be specified and thus potentially increasing accuracy. Watershed would have benefited from improved seed pixel placement, though this would require some kind of preliminary segmentation in order to find the best seed position. Quadtrees could have had a merge component added, which would have reduced the number of segments in the final segmentation. This would have allowed us to reach a high level of accuracy in much fewer segments.

There are many segmentation types that were not examined here, such as model based methods or hybrid methods. Future work could focus on further ruling out "bad" segmentation algorithms, and try to find an even more effective algorithm than our clustering methods. However, due to the satisfactory results already demonstrated here, there would possibly be little to gain in doing this.

Further optimisation of our mean-shift implementation for vine images could be performed, especially in terms of segmentation run-time. The implementation used here were nowhere near fast enough for real-time use, so speeding it up would be most useful for end-system use in the robot.

The next task in the processing pipeline of our robot is to classify each of the segments in an image, assigning it a label such as vine, post, wire, etc., using model-based classification methods [9]. Once all vine segments have been identified, a model of the vine can be constructed, from which we can determine the best location to cut. There are several other researchers currently working on these problems for use in the robot.

Bibliography

[1] MATLAB Normalized Cuts Segmentation Code, November 2012. `http://www.cis.upenn.edu/ jshi/software/`.

[2] Quad-Tree segmentation - MEX - File Exchange - MATLAB Central, November 2012. `http://www.mathworks.com/matlabcentral/fileexchange/27993-quad-tree-segmentation-mex`.

[3] C. H. A. Grote, M. Butenuth. Road extraction in suburban areas based on normalized cuts. *ISPRS PIA07, Photogrammetric Image Analysis*, 2007.

[4] U. Bacher and H. Mayer. Automatic road extraction from multispectral high resolution satellite images. 2005.

[5] M. Balafar, A. Ramli, M. Saripan, and S. Mashohor. Review of brain mri image segmentation methods. *Artificial Intelligence Review*, 33(3):261–274, 2010.

[6] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. Mathematical morphology in image processing. *Optical Engineering*, 34:433–481, 1993.

[7] A. Bibi, A. Khan, and M. S. H. Khiyal. Modified watershed algorithm for segmentation of 2d images. *Issues in Informing Science and Information Technology*, 6:877+, 2009.

[8] T. Botterill, S. Mills, and R. Green. Design and calibration of a hybrid computer vision and structured light 3d imaging system. In *In Proceedings of the International Conference on Automation, Robotics, and Applications (ICARA)*, 2011.

[9] T. Botterill, S. Mills, and R. Green. Reconstructing partially visible models using stereo vision, structured light, and the g2o framework. In *In Proceedings of Image and Vision Computing New Zealand*, 2012.

[10] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112 vol.1.

[11] F. Camastra and A. Vinciarelli. *Clustering Methods*, pages 117–148. Springer London, London, 2008.

[12] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[13] J. M. Flores and F. Schmitt. Segmentation, reconstruction and visualization of the pulmonary artery and the pulmonary vein from anatomical images of the visible human project. In *Computer Science, 2005. ENC 2005. Sixth Mexican International Conference on*, pages 136–144.

[14] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[15] J. Hang, F. Yanming, and L. Bofeng. Road network extraction with new vectorization and pruning from high-resolution rs images. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6, 2008.

[16] R. Hidayat. Texture-boundary detection in real-time. 2010.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.

[18] O. Javed and M. Shah. Tracking and object classification for automated surveillance. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 439–443. Springer Berlin / Heidelberg, 2006.

[19] H. Jia and P. Tang. An improved fingerprint image segmentation algorithm. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 2, pages 118–120, 2012.

[20] S. Jianbo and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[21] R. Kasturi, R. Jain, and B. G. Schunck. *Machine vision*. McGraw-Hill, New York, 1995.

[22] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *Proceedings of the 11th European conference on Computer vision: Part II*, ECCV'10, pages 480–493, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] Z. Li and L.-q. Huang. A new segmentation method of cr images based on discrete wavelet transform and mathematics morphology. pages 60273H–60273H–6, 2006.

[24] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129 – 137, mar 1982.

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[26] H. Mayer, S. Hinz, U. Bacher, and E. Baltsavias. A test of automatic road extraction approaches. 2006.

[27] N. J. B. McFarlane, B. Tisseyre, C. Sinfort, R. D. Tillett, and F. Sevila. Image analysis for pruning of long wood grape vines. *Journal of Agricultural Engineering Research*, 66(2):111–119, 1997.

[28] J. Naugle, G. Rehkugler, and A. S. o. A. E. S. Meeting. *Grapevine Cordon Following Using Digital Image Processing*. American Society of Agricultural Engineers, 1987.

[29] Z. Qian. Model-based image segmentation in medical applications, 2007.

[30] O. Samko, Y.-K. Lai, D. Marshall, and P. Rosin. Segmentation of parchment scrolls for virtual unrolling. *Proc. BMVC*, 2011.

[31] S. Sapna Varshney, N. Rajpal, and R. Purwar. Comparative study of image segmentation techniques and object matching using segmentation. In *Methods and Models in Computer Science, 2009. ICM2CS 2009. Proceeding of International Conference on*, pages 1–6, 2009.

[32] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Computer Vision, 1998. Sixth International Conference on*, pages 1154 –1160, jan 1998.

[33] M. Singh, P. Patel, D. Khosla, and T. Kim. Segmentation of functional mri by k-means clustering. *Nuclear Science, IEEE Transactions on*, 43(3):2030–2036, 1996.

[34] Z. Song, N. Tustison, B. Avants, and J. Gee. *Integrated Graph Cuts for Brain MRI Segmentation Medical Image Computing and Computer-Assisted Intervention MICCAI 2006*, volume 4191 of *Lecture Notes in Computer Science*, pages 831–838. Springer Berlin / Heidelberg, 2006.

[35] S. Tatiraju and A. Mehta. Image segmentation using k-means clustering, em and normalized cuts. 2008.

[36] L. Tianhu, J. K. Udupa, P. K. Saha, and D. Odhner. Artery-vein separation via mra-an image processing approach. *Medical Imaging, IEEE Transactions on*, 20(8):689–703, 2001.

[37] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):929–944, 2007.

[38] H. Zhang, J. E. Fritts, and S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280, 2008.

[39] Q. Zhang and I. Couloigner. Automated road network extraction from high resolution multi-spectral imagery. 2006.